

แบบฝึกหัดเรื่อง stack

1. (1_postfix_eval) รับค่านิจนแบบ postfix notation ที่ประกอบด้วยตัวดำเนินการบวก ลบ คูณ หาร โดยใช้เครื่องหมาย '+' '-' '*' และ '/' ให้คำนวณค่านิจนที่ได้ โดยใช้ stack operation ข้อมูลนำเข้าประกอบด้วยจำนวนเต็มและตัวดำเนินการ เว้นด้วยช่องว่างหนึ่งช่อง มีตัวอย่างการแสดงผลดังนี้

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
5 2 3 + /	1

2. (2_infix_eval) รับค่านิจนแบบ infix notation ที่ประกอบด้วยบวก ลบ คูณ หาร และวงเล็บ โดยใช้เครื่องหมาย '+' '-' '*' และ '/' ให้คำนวณค่านิจนที่ได้ โดยใช้ stack operation ข้อมูลนำเข้าประกอบด้วยจำนวนเต็มและตัวดำเนินการเขียนติดกัน มีตัวอย่างการแสดงผลดังนี้

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
(2 * ((2 + 3) + 4))	18

หมายเหตุ ให้ใช้คำสั่งในการรับค่าสตริงได้ดังนี้

```
string exp;
getline(cin, exp);
```

(2 * (2 + 3))

str

st[

st = (* + b) (แล้วก็เซต)

stack num
↓
2 2 3

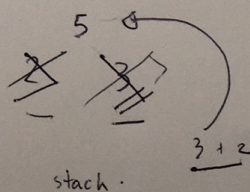
stack opr
* .

stack .

stack

stack num
↓
2, 2, 3

stack ope
↓
(* (+



3. (3_postfix_infix) ให้เขียนโปรแกรมเพื่อแปลงนิพจน์แบบ postfix notation ให้เป็น infix notation ที่ประกอบด้วยตัวดำเนินการ การบวก ลบ คูณ หาร และวงเล็บ โดยใช้เครื่องหมาย '+', '-', '*', '/', '(', '(', และ ')' ส่วนการใส่วงเล็บ จะใส่วงเล็บเมื่อต้องการจัดลำดับของการคำนวณเท่านั้น เช่น $a + b * c$ การทำงานของ $b * c$ จะทำก่อน $a + b$ ดังนั้น ไม่จำเป็นต้องใส่วงเล็บให้กับ $b * c$ มีตัวอย่างการแสดงผลดังนี้

X

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
a b c * + d e * f + g * +	a + b * c + (d * e + f) * g

4. (4_infix_postfix) ให้เขียนโปรแกรมเพื่อแปลงนิพจน์แบบ infix notation ให้เป็น postfix notation ที่ประกอบด้วยตัวดำเนินการ การเช่นเดียวกับข้อ 3 มีตัวอย่างดังนี้

ไม่ใส่วงเล็บ

X

ตัวอย่างข้อมูลนำเข้า	ตัวอย่างข้อมูลส่งออก
a + b * c + (d * e + f) * g	a b c * + d e * f + g * +